UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Žiga Cigole

Mobilna aplikacija za rezervacijo sedeža na vlaku

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2017

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Žiga Cigole

Mobilna aplikacija za rezervacijo sedeža na vlaku

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Aljaž Zrnec

Ljubljana, 2017

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Vlak sodi med enega izmed bolj uporabljenih javnih prevoznih sredstev. Zaradi tega se večkrat zgodi, da so vlaki prenatrpani. V okviru diplomske naloge razvijte mobilno aplikacijo, ki bo potnikom omogočila predhodno rezervacijo sedeža na vlaku ter s tem zagotovila prosto mesto. V okviru diplomskega dela tudi predstavite vse uporabljene tehnologije in orodja, uporabljena za razvoj take aplikacije ter komentirajte ugotovitve.

Zahvaljujem se predvsem svojemu mentorju viš. pred. dr. Aljažu Zrnecu za vso pomoč ter nasvete, s katerimi mi je pomagal pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic

Kratica	Angleško	Slovensko
WORA	Write once, run anywhere	Spiši enkrat, poženi kjerkoli
API	Application programming interface	Vmesnik za namensko programiranje
SCP	Secure copy	Varno kopiranje
SSH	Secure shell	Varna lupina
SFTP	Secure file transfer protocol	Protokol za varen prenos datotek
LAMP	Linux, Apache, MySQL, PHP	Linux, Apache, MySQL, PHP
SQL	Structured query language	Strukturirani povpraševalni jezik
ER	Entity-relationship model	Entitetno-relacijski model
HTML	Hyper text markup language	Hipertekstni označevalni jezik
PHP	Hypertext pre-processor	Hipertekstni preprocesor
ID	Identification	Identifikator
URL	Uniform resource locator	Enolični krajevnik vira
CSS	Cascading style sheets	Kaskadne stilske podloge
DOM	Document object model	Dokumentni objektni model
LP, LPV	Passenger train	Potniški vlak
ICS	InterCity Slovenia	InterCity Slovenija
WIFI	Wireless Fidelity	Brezžična povezava

Povzetek

Naslov: Mobilna aplikacija za rezervacijo sedeža na vlaku

Mobilne aplikacije v današnjem času niso več nič posebnega, uporabljajo se vsakodnevno, praktično na vsakem koraku. Medtem ko je v večini evropskih držav to že uveljavljeno, pri nas še ne obstaja aplikacija za rezervacijo sedežev na vlaku. Tako smo si zadali cilj, razviti aplikacijo za mobilni operacijski sistem, ki bo le-to omogočala. V diplomskem delu so predstavljene tehnologije in orodja, ki so bila uporabljena za izdelavo mobilne aplikacije Android. Prav tako je opisan proces razvoja mobilne aplikacije, vključno z vsemi ostalimi komponentami, kot so strežnik, podatkovna baza ter spletne storitve. V zaključku dela je predstavljena faza testiranja in pojasnjene sklepne ugotovitve.

Ključne besede: mobilne aplikacije, Android, rezervacija sedeža, vlak

Abstract

Title: A mobile application for booking a seat on the train

Mobile applications in the present day are nothing special anymore, they are used daily practically at every step. While this is already established in most of the European countries, we still don’t have an application for booking a seat on the train. Therefore, our goal was to develop an application for a mobile operating system which will enable this. This thesis covers the technologies and tools that were used in the making of our Android mobile application. The process of mobile application development is also described, including other components such as server, database and web services. At the end of our thesis we cover the test phase and conclusions.

Keywords: mobile applications, Android, seat reservation, train

Poglavje 1 Uvod

Pogosto se zgodi, da potniki ujamejo vlak šele nekaj minut pred njegovim odhodom, saj jim drugače ne dopušča služba ali šola. Posledično morajo nato med vožnjo celo pot stati, saj so vsi sedeži že zasedeni, kar pa je neudobno in utrujajoče. Da bi potnikom olajšali omenjene težave, je bila v okviru diplomskega dela razvita Android aplikacija, ki omogoča rezervirati sedež na vlaku. Ponuja namreč možnost, da si potnik preko pametnega telefona predhodno zagotovi sedež, kar mu omogoča prijetno in udobno potovanje.

1.1 Ozadje

Pametni telefoni in naprave so že dalj časa del vsakdanjega življenja. Število uporabnikov iz dneva v dan narašča, kar potrjuje tudi statistika, ki pravi, da že več kot 75 % svetovne populacije uporablja pametno mobilno napravo vsak dan. Zaradi tega je razvoj mobilnih aplikacij postal zelo dobičkonosen posel. Aplikacij ne razvijajo samo velika podjetja, ampak tudi posamezniki, ker razvoj enostavne mobilne aplikacije ne predstavlja več večjega problema, saj se vsa ustrezna literatura, enostavni vodiči in video prikazi lahko pridobijo na spletu.

1.2 Namen in prispevek diplomskega dela

Ideja za diplomsko delo se je porodila iz dejstva, da večina železnic v tujini, ki se ukvarjajo z javnim prevozom, že imajo svoje mobilne aplikacije, ki uporabniku olajšajo potovanje, medtem ko pri nas tovrstna aplikacija za rezervacijo kart na vlaku še ne obstaja. Glavni izziv je bil torej razviti mobilno aplikacijo, ki bo potnikom ponujala rezervacijo sedeža na vlaku. Prav tako je bilo treba zagotoviti hrambo vseh podatkov glede rezervacij na strežniku.

1.3 Struktura diplomskega dela

V drugem poglavju diplomskega dela se srečamo z opisom tehnologij in orodij, ki smo jih uporabili pri izdelavi aplikacije. V nadaljevanju je predstavljen razvoj, vse od same mobilne aplikacije do implementacije podatkovne baze in spletnih storitev na aplikacijskem strežniku. V četrtem poglavju so opisane vrste in rezultati testiranj, v zaključku dela pa podamo še sklepne ugotovitve in predloge za izboljšanje aplikacije.

Poglavje 2 Uporabljene tehnologije in orodja

Omenjeno poglavje opisuje tehnologije in orodja, ki so bila uporabljena v okviru diplomskega dela za razvoj mobilne aplikacije za operacijski sistem Android.

2.1 Android

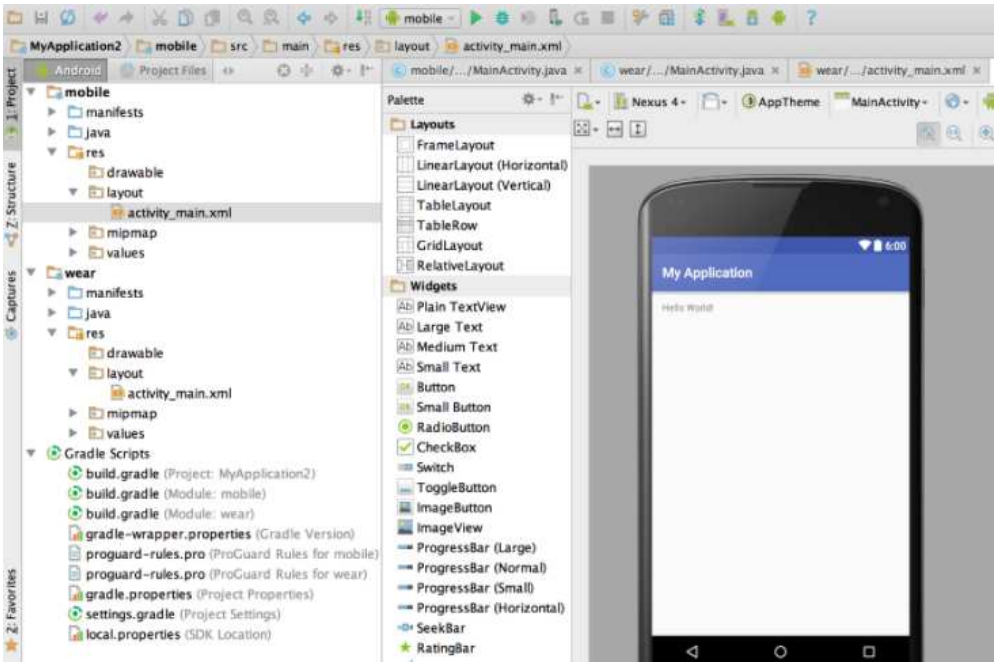
Android je mobilni operacijski sistem, ki ga je razvilo podjetje Google [5]. Temelji na operacijskem sistemu Linux [9] in je bil primarno razvit za mobilne aplikacije z ekranom na dotik (ang. touchscreen), kot so pametni telefoni in tablice. Njegov grafični vmesnik deluje na podlagi neposredne manipulacije s pomočjo gest na dotik (tipkanje, drgnjenje s prstom po zaslonu). Prvič je bil predstavljen leta 2007, od 2008 pa se uporablja tudi po precej komercialnih Android napravah. Doživel je že veliko nadgradenj, prva različica se je imenovala Android 1.0, trenutna pa ima oznako Android 8 – Oreo. Od maja 2017 ima Android dve milijardi aktivnih mesečnih uporabnikov s tem pa tudi največje število naprav, na katerih teče.

Sama izvorna koda Androida je bila izdana z odprtokodnim dovoljenjem, kar pomeni, da je pod določenimi pogoji na voljo javnosti za uporabo, urejanje in deljenje [3]. Zaradi tega je razvoj programov znatno cenejši in lažji. To občutijo predvsem uporabniki, saj so programi za operacijski sistem Android večinoma brezplačni za uporabo.

2.1.1 Android Studio

Je uradno okolje za razvoj aplikacij, namenjenih operacijskemu sistemu Android [1]. Android Studio je brezplačen in na voljo vsem uporabnikom [17]. Služi kot zamenjava za razvojno okolje Eclipse, saj nudi podporo za razvojna orodja Android in ima preglednejši uporabniški vmesnik. Okolje je bilo predstavljeno leta 2013, prva stabilna različica pa je izšla decembra 2014. Android Studio z vsako novo različico podpira vedno več funkcij, ki uporabniku olajšajo razvoj. Nekatere izmed njih so: inteligentni urejevalnik kode, optimizacija za različne Android naprave, vnaprej pripravljene predloge in orodja za testiranje. Pri izdelavi

diplomskega dela je bila uporabljena različica Android Studio 2.2.3. Slika 1 prikazuje grafični vmesnik razvojnega okolja Android Studio.



Slika 1: Razvojno okolje Android Studio

2.1.2 Java

Java je moderen, objektno orientiran programski jezik [10]. Namenjen je programiranju v načinu WORA (ang. Write Once, Run Anywhere), kar dejansko pomeni, da se prevedena Java koda lahko izvaja na različnih platformah, ki podpirajo Javo, brez potrebe po ponovnem prevajanju. To poteka tako, da se Java izvorna koda prevede v format bytecode, nato pa navidezni stroj Java (ang. Java Virtual Machine) poskrbi za izvedbo. Sama sintaksa programskega jezika precej spominja na C in C++. Trenutno je Java eden izmed popularnejših programskih jezikov. Najde se jo na veliko različnih napravah, na pametnih telefonih, osebnih računalnikih in tudi super računalnikih. Jezik Java predstavlja uradni jezik za razvoj Android aplikacij, hkrati pa je tudi večji del samega operacijskega sistema Android napisan v Javi, prav tako njegovi programski vmesniki.

2.2 Linux

Je računalniški operacijski sistem, ki temelji na sistemu UNIX. Linux izvorna koda je prosto dostopna [18]. Primarno je bil razvit za uporabo na osebnih računalnikih z Intel x86 arhitekturo, vendar je zaradi svoje prilagodljivosti bil prenešen na več platform kot katerikoli

drug operacijski sistem. Linux je večuporabniški (podpira hkratno delo več uporabnikov) in večopravilen (podpira hkratno izvajanje več opravil) operacijski sistem.

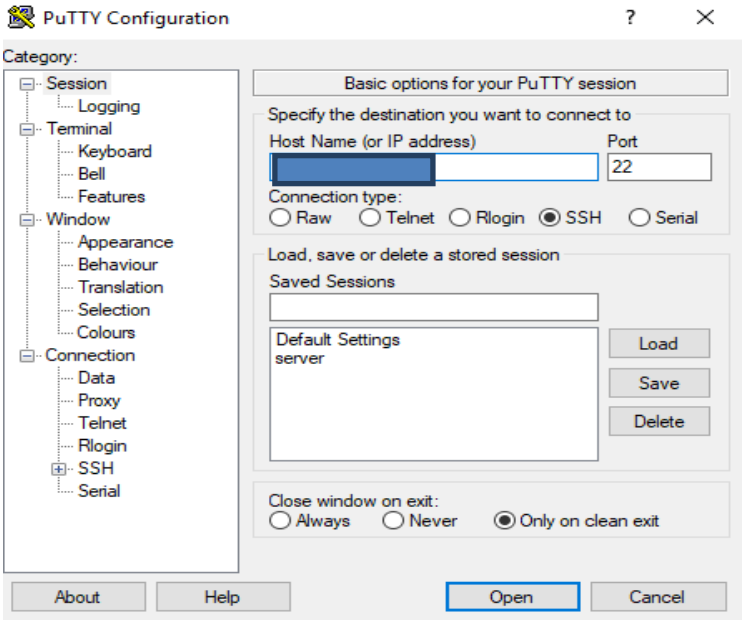
Danes poznamo več izdaj – distribucij operacijskega sistema Linux. Nekatere izmed najbolj znanih so: CentOS, Debian, Linux Mint, Ubuntu in Red Hat. Vsaka od Linux distribucij vključuje Linux jedro z vsemi potrebnimi knjižnicami ter pripomočki za konfiguracijo operacijskega sistema. Prav tako distribucija vključuje večje število aplikacij, tako odprtokodnih kot lastniških. Za izdelavo diplomskega dela je bila uporabljena distribucija Ubuntu, različice Ubuntu 16.04.2.

2.2.1 Putty

Putty je odprtokodni emulator terminala [11], serijska konzola in aplikacija za prenos datotek preko interneta. Na voljo je za Windows in Unix platforme. Nudi podporo precej internetnim protokolom vključno s SCP, SSH in Telnet. Samo ime Putty nima posebnega pomena. Putty je bil razvit v programskem jeziku C, prvič pa predstavljen leta 1999. Uporabnikom omogoča oddaljen dostop do računalnikov preko interneta. Je zelo popularno orodje za tekstovno komunikacijo med napravami in za povezovanje na Linux strežnike preko Microsoft operacijskih sistemov. Nekatere njegove pomembnejše funkcije so:

- podpora Unicode,
- podpora IPv6,
- avtentikacija z javnim ključem,
- kontrola SSH šifriranega ključa in različice protokola in
- prenos datotek s pomočjo SCP in SFTP odjemalcev.

Za izdelavo diplomskega dela je bila uporabljena različica programa Putty 0.68. Slika 2 prikazuje uporabniški vmesnik orodja Putty.



Slika 2: Uporabniški vmesnik orodja Putty

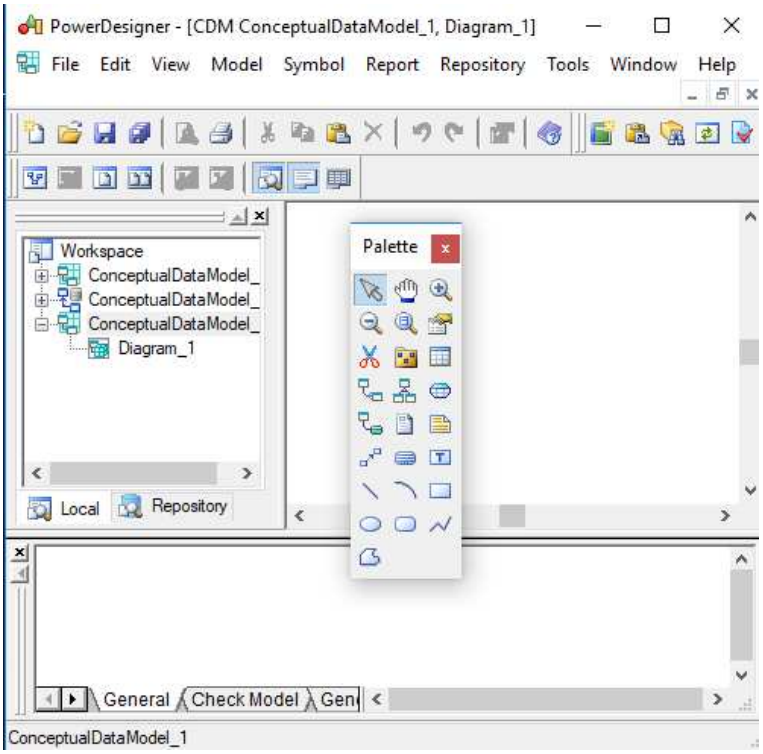
2.3 MySQL

MySQL je odprtokodni relacijski sistem za upravljanje s podatkovnimi bazami, ki za delo uporablja jezik SQL [6]. SQL je kratica za Structured Query Language in predstavlja najbolj uporabljen jezik, ki se ga uporablja za dostop do relacijskih zbirk podatkov. MySQL je osrednja komponenta odprtokodnega programskega sklada LAMP, ki je v bistvu akronim za zbirko programske opreme: Linux, Apache, MySQL in Perl/PHP/Python [14]. Uporablja se pri najbolj poznanih spletnih platformah, kot so Google, Facebook, Twitter in YouTube. Napisan je s pomočjo programskih jezikov C in C++, medtem ko je njegov razčlenjevalec napisan v yacc-u (program, ki s pomočjo prej določenih skladenjskih pravil vrne strukturirano kodo) [19]. MySQL deluje po modelu odjemalec-strežnik. Ta model omogoča, da odjemalec in strežnik med seboj komunicirata po vzorcu zahteva-odgovor, kar pomeni, da odjemalec pošlje zahtevo, strežnik pa vrne odgovor.

Prva različica, ki ni imela nekega posebnega naziva, se je pojavila leta 1995, trenutno najsodobnejša pa je različica MySQL 5.7.19, ki je bila tudi uporabljena pri izdelavi diplomskega dela.

2.3.1 PowerDesigner

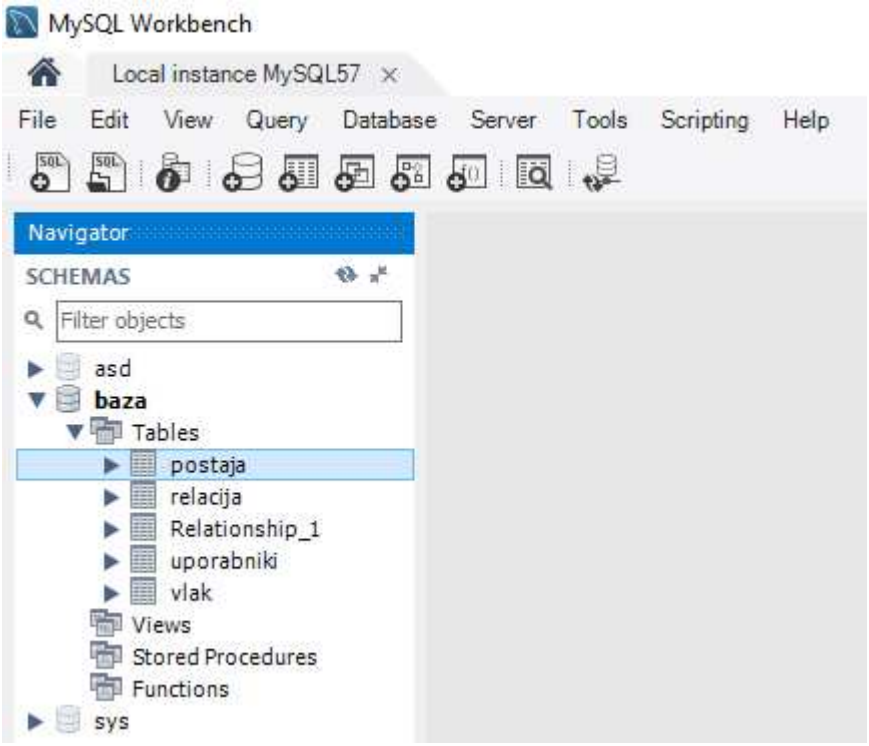
PowerDesigner je CASE (ang. Computer Aided Software Engineering) orodje [4], ki je preprosto za uporabo in omogoča integrirano modeliranje (povezovanje posameznih modelov med seboj glede na njihove odvisnosti) podatkovnih baz z uporabo standardnih metodologij [13]. Uporabnikom omogoča, da enostavneje analizirajo, vizualizirajo in upravljajo podatke o podatkih (metapodatke). Prva različica, ki se je imenovala AMC*Designor [19], je bila namenjena grajenju Oracle podatkovnih baz, kmalu pa so dodali podporo tudi vsem večjim relacijskim sistemom za upravljanje s podatkovnimi bazami. PowerDesigner ponuja tudi možnost objektnega modeliranja, generiranja poročil in avtomatskega generiranje kode SQL, JAVA in .NET. Za izdelavo diplomskega dela je bila uporabljena različica 12.5. Slika 3 prikazuje uporabniški vmesnik orodja PowerDesigner.



Slika 3: Uporabniški vmesnik orodja PowerDesigner

2.3.2 MySQL Workbench

Je brezplačno orodje za razvoj podatkovnih baz, ki ga uporabljajo arhitekti in administratorji podatkovnih baz ter programerji [7]. Samo razvojno okolje združuje SQL razvoj, administracijo, načrtovanje podatkovne baze, ustvarjanje in vzdrževanje le te. Je naslednik orodja DBDesigner 4. Ponuja grafična orodja za ustvarjanje, izvedbo in pomoč pri optimizaciji SQL poizvedb ter nudi vizualno konzolo, ki omogoča lažjo administracijo MySQL okolja, s tem pa tudi precej poenostavljen pogled na podatkovno bazo. Prva različica je izšla leta 2005 in je imela oznako MySQL Workbench 5.0, trenutno najnovejša, ki je bila tudi uporabljena za izdelavo diplomskega dela, pa nosi oznako 6.3. Slika 4 prikazuje uporabniški vmesnik orodja MySQL Workbench.



Slika 4: Uporabniški vmesnik orodja MySQL Workbench

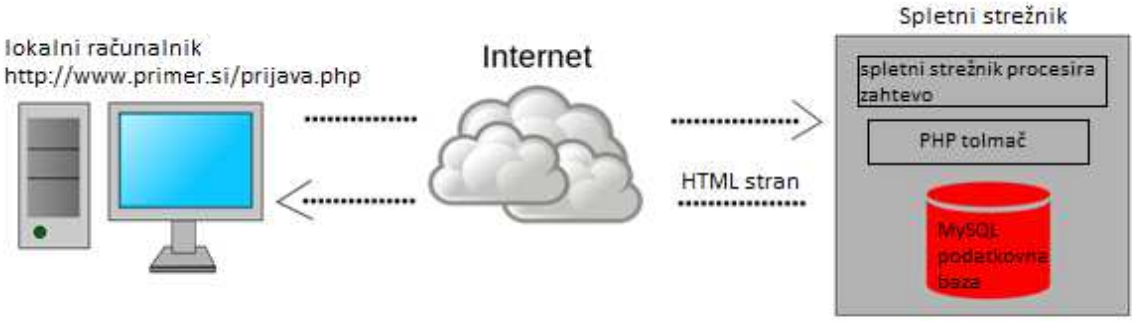
2.4 Notepad++

Notepad++ je urejevalnik besedil in programske kode za uporabo v operacijskem sistemu Microsoft Windows [2]. Omogoča delo z več hkrati odprtimi datotekami znotraj enega okna. Razvit je bil v programskem jeziku C++ in se ga distribuira kot brezplačno programsko opremo. Predstavljen je bil leta 2003, od takrat pa je prejel že mnogo nagrad in pohval kot najboljši Windows urejevalnik besedil, ki je namenjen programiranju. Temu so močno pripomogle njegove glavne funkcije, ki olajšajo programiranje, saj nudijo poudarjanje

sintakse z barvami, zlaganje kode (uporabnik lahko skrije določen del kode, da ima boljši pregled) ter tudi avtomatsko dokončevanje besed za programerske in označevalne jezike. Novejše različice nudijo podporo številnim programskim jezikom, kot so C, C#, C++, Fortran, HTML, Java, JavaScript, PHP, Python in še mnogim drugim. Za izdelavo diplomskega dela je bila uporabljena različica 7.3.2.

2.5 PHP

PHP je skriptni jezik, ki je primarno namenjen spletnemu razvoju, vendar se ga uporablja tudi kot splošno namenski programski jezik [16]. Na začetku je kratica PHP predstavljala Personal Home Page, sedaj pa so te tri črke rekurzivni akronim (akronim, ki se nanaša sam nase v izrazu, ki ga podaja) za Hypertext Preprocessor. PHP koda je lahko vgrajena v samo HTML kodo ali pa se jo uporablja z različnimi sistemi za spletne predloge in sistemi za upravljanje spletnih strani. PHP koda se sprocesira s pomočjo interpreta oz. tolmača, ki je implementiran kot modul na spletnem strežniku. Programska oprema na strežniku prejme PHP izvorno kodo kot vhod in nato generira spletno stran kot izhod, kar prikazuje tudi slika 5. Je brezplačen za namestitev in uporabo. Prva različica je izšla leta 1995, trenutno najnovejša pa ima oznako PHP 7.1.8. Za izdelavo diplomskega dela je bila uporabljena različica 5.6.25.



Slika 5: Prikaz poteka komunikacije med računalnikom in tolmačem na strežniku

Slika 6 predstavlja PHP program, ki je vgrajen v HTML dokument.

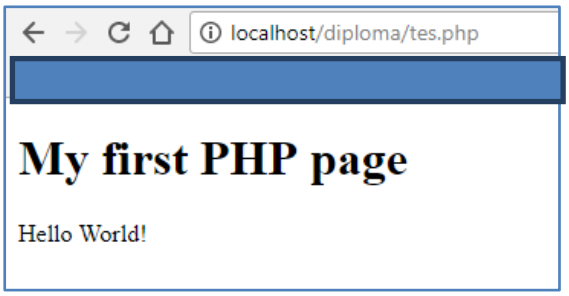
```
<!DOCTYPE html>
<html>
    <body>
    <h1>My first PHP page</h1>

    <?php
    echo "Hello World!";
    ?>

    </body>
</html>
```

Slika 6: PHP program

Če bi zgoraj omenjeni program iz slike 6 testirali na osebnem računalniku, ki ima nameščeno ustrezno programsko opremo, kot je WAMP (ang. Windows, Apache, MySQL, PHP), bi rezultat programa bil spletna stran z glavo »My first PHP page« ter napisom »Hello World!«. To tudi prikazuje slika 7.



Slika 7: Izpis programa

Poglavje 3 Razvoj aplikacije

V tem poglavju je predstavljena pot od ideje do razvoja aplikacije in kako smo se lotili načrtovanja. Sledi opis razvoja vseh komponent, ki sestavljajo našo aplikacijo. Te komponente so: mobilna Android aplikacija, strežnik, podatkovna baza in spletne storitve.

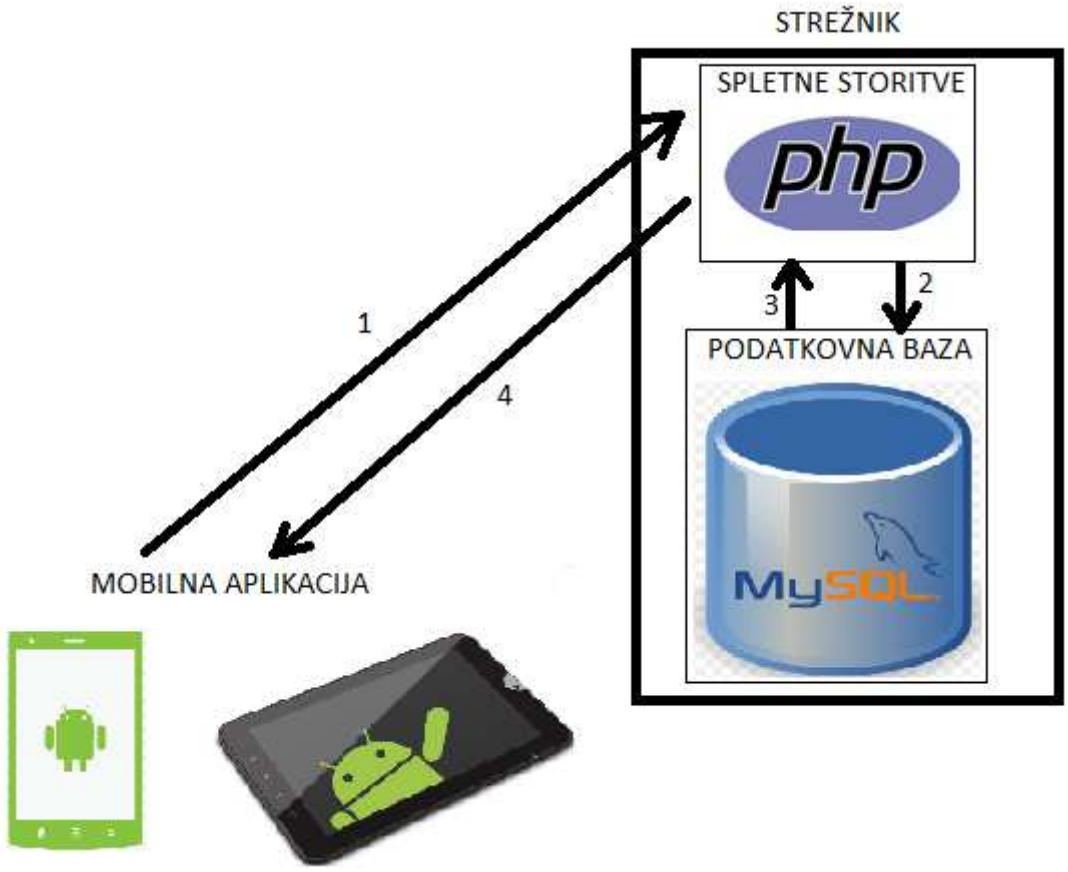
3.1 Ideja

Ideja za to diplomsko delo je bila razviti mobilno aplikacijo, ki bo omogočala rezervacijo sedeža na vlaku. Treba je bilo tudi poskrbeti za hrambo podatkov o rezervacijah in omogočiti, da bodo na voljo za kasnejšo analizo.

Odločili smo se, da našo aplikacijo razvijemo za operacijski sistem Android, ker je ta trenutno najbolj razširjen mobilni operacijski sistem na trgu. Prav tako smo z razvojem aplikacij za ta operacijski sistem že imeli nekaj izkušenj. Za Android smo se odločili tudi zato, ker smo za potrebe testiranja aplikacije lahko dobili največ možnih kandidatov.

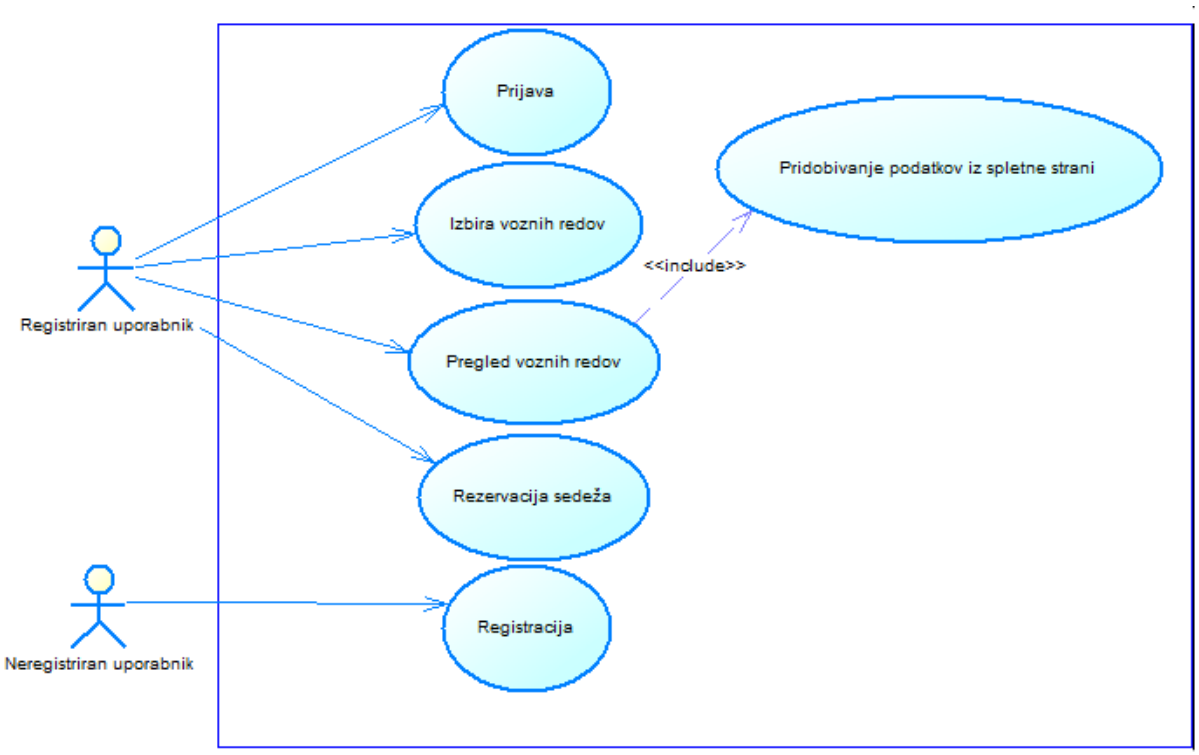
3.2 Načrtovanje

Ker je bilo od začetka precej nejasno, kako naj bi aplikacija izgledala in delovala, smo se najprej lotili načrtovanja grafičnega vmesnika. Grafični vmesnik mobilne aplikacije smo skicirali na list papirja, da smo si zadevo lažje predstavljali. Narisali smo tudi model, ki prikazuje povezave med vsemi komponentami mobilne aplikacije. Slika 8 prikazuje omenjen model, ki ga sestavljajo mobilna aplikacija, strežnik, podatkovna baza in spletne storitve, ki so potrebne za povezovanje s podatkovno bazo.



Slika 8: Model povezav med komponentami

Po grafičnem vmesniku je bilo treba določiti funkcionalnosti mobilne aplikacije. Prva funkcionalnost aplikacije je prijava uporabnika. Če uporabnik nima svojega uporabniškega imena ter gesla, se mora sprva registrirati in zatem se lahko prijavi. Sledi izbira vstopne postaje, izstopne postaje in datuma, za kateri dan želi opraviti rezervacijo sedeža. Odločili smo se, da mora aplikacija preko javno dostopne spletne strani (http://www.slo-zeleznice.si/sl/) pridobivati podatke o voznih redih. Potem sledi prikaz vseh možnih voznih redov za izbran datum in relacijo, nato pa je treba izbrati še želeno število prostih sedežev na vlaku. Za to poskrbijo spletne storitve, s pomočjo katerih aplikacija dostopa do podatkovne baze. Podatkovna baza je na našem strežniku, hrani pa podatke o uporabnikih, vlakih ter rezervacijah. Slika 9 prikazuje diagram primerov uporabe mobilne aplikacije.



Slika 9: Diagram primerov uporabe

Na strežniku je nameščen operacijski sistem Linux. Namen strežnika je, da na njem teče podatkovna baza, na kateri se shranjujejo in urejajo vsi podatki, ki so potrebni za pravilno delovanje mobilne aplikacije. Prav tako se na strežniku izvajajo spletne storitve, ki skrbijo za vzpostavljanje povezave med mobilno aplikacijo in podatkovno bazo. Zaradi večje zanesljivosti in boljše dostopnosti smo namesto lokalnega uporabili oddaljeni strežnik.

Pri izdelavi podatkovne baze smo sprva izdelali logični podatkovni model v orodju PowerDesigner, nato pa model pretvorili v SQL kodo, s pomočjo katere smo nato generirali podatkovno bazo.

3.3 Mobilna aplikacija

Po končanem načrtovanju smo se lotili razvoja mobilne aplikacije. Mobilna Android aplikacija je namenjena slovenskim uporabnikom, zato se za vse gradnike v aplikaciji uporabljajo slovenski napisi. Da aplikacija pravilno deluje, je potreben dostop do interneta s pomočjo mobilnih podatkov ali brezžične WIFI tehnologije. Dostop do interneta omogoča povezovanje na strežnik in dostopanje do podatkov. Brez mobilne povezave uporaba aplikacije ni mogoča. Aplikacija vsebuje 5 aktivnosti tipa .xml, ki predstavljajo grafični

vmesnik in 6 razredov tipa .java, ki vsebujejo programsko kodo, ki skrbi za delovanje aplikacije. V nadaljevanju so podani opisi aktivnosti, pri katerih navedemo njihove gradnike, ter opisi razredov, kjer opišemo, čemu je posamezen razred dejansko namenjen.

3.3.1 Opis aktivnosti

Aktivnost je ena izmed temeljnih gradnikov vsake aplikacije na Android platformi. Služi kot vstopna točka za uporabnikov stik z aplikacijo, predstavljena pa je v obliki zaslonske maske. Aplikacija ima lahko več aktivnosti, nujno pa mora vsebovati vsaj eno, ki predstavlja začetno stran. Aktivnost lahko požene drugo aktivnost, prav tako pa ima vsaka aktivnost svoj življenjski cikel, katerega upravljajo posebne metode: onCreate(), onStart(), onResume(), onPause(), onStop() in onDestroy().

Vsi gradniki, ki obstajajo v Android aplikaciji, morajo biti najprej deklarirani v konfiguracijski datoteki manifest (AndroidManifest.xml). Omenjena datoteka zagotavlja operacijskemu sistemu bistvene informacije o aplikaciji, brez katerih se aplikacijska koda ne požene. Nekatere informacije, ki jih vsebuje, so: najmanjša verzija programskega vmesnika (v primeru, da ima uporabnikova naprava manjšo različico programskega vmesnika, kot je določeno, mu Android sistem ne bo dovolil namestitve aplikacije) ter različica in ime programske kode. Med drugim je v datoteki manifest poskrbljeno tudi za dovoljenja, ki jih bo aplikacija potrebovala, kar se nanaša na dostop do podatkov v napravi in do spleta ter deklaracije zunanjih API knjižnic.

3.3.1.1 Aktivnost main

Aktivnost »main« predstavlja v naši aplikaciji prvo zaslonsko masko, ki se odpre ob zagonu aplikacije na mobilni napravi in je prva izmed obveznih aktivnosti.

Prva zaslonska maska v našem primeru vsebuje 6 gradnikov, definiranih v datoteki manifest:

- 2 krat TextView
    - Uporabniško ime
    - Geslo
- 2 krat EditText
    - Polje za vnos uporabniškega imena
    - Polje za vnos gesla

- 2 krat Button
    - Prijava
    - Registracija

Vse aktivnosti so bile sestavljene s pomočjo gradnika RelativeLayout, ki omogoča gradnikom v omenjeni postavitvi, da se jim nastavi položaj relativno na njihov starševski element. Slika 10 predstavlja del programske kode aktivnosti »main«, razberemo pa lahko, da je gradnik Button z ID-jem »gumb_prijava« postavljen pod element »textPassword«.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="64dp"
    android:paddingRight="64dp"
    android:paddingTop="16dp"
    tools:context="com.example.userasus.sz.MainActivity">

    <Button
        android:text="Prijava"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textPassword"
        android:layout_alignLeft="@+id/textPassword"
        android:layout_alignStart="@+id/textPassword"
        android:layout_marginTop="19dp"
        android:id="@+id/gumb_prijava"
        android:onClick="prijava"/>
```

Slika 10: Koda XML, ki definira relativno postavitev in gradnik Button

Razberemo lahko tudi, da je besedilo tega gradnika »Prijava«, njegov ID je »gumb_prijava« in metoda, ki se izvede ob kliku na ta gumb, se imenuje »prijava«.

Slika 11 predstavlja zaslonsko masko za »aktivnost main«.

Slika 11: Zaslonska maska aktivnosti »main«

3.3.1.2 Aktivnost registracija

Ta aktivnost je opcijska, kar pomeni, da se uporabi oz. prikaže samo v primeru, ko se hoče nov uporabnik registrirati. Torej, ob pritisku na »gumb_registracija« na aktivnosti »main« se s pomočjo programske kode prikaže aktivnost »registracija«. V tem primeru mora uporabnik vnesti podatke, kot so: uporabniško ime, geslo in nato še ponovitev istega gesla, da ne pride do napake pri prvem vnosu. Sestavni deli aktivnosti »registracija« so:

- 3 krat TextView
    - Uporabniško ime
    - Geslo
    - Ponovi geslo
- 3 krat EditText
    - Polje za vnos uporabniškega imena
    - Polje za vnos gesla
    - Polje za ponovitev gesla

- 1 krat Button
    - Registriraj

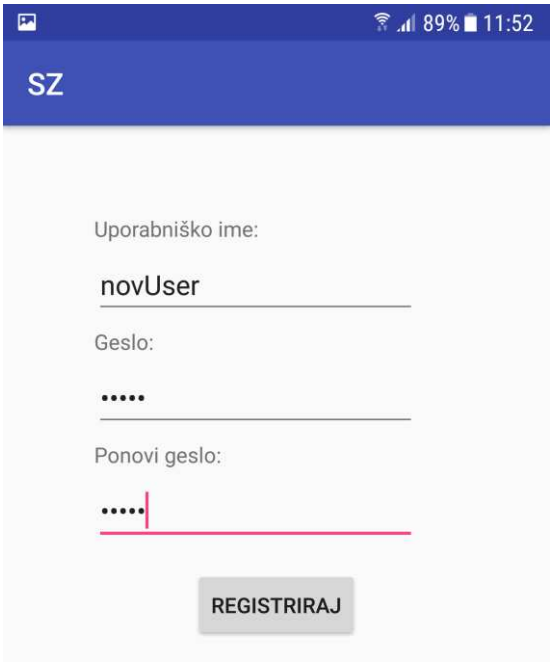
Slika 12 predstavlja programsko kodo, ki definira gradnik TextView.

```
<TextView
    android:text="Ponovi geslo:"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:layout_below="@id/textPasswordReg"
    android:id="@+id/textView6"
    android:layout_alignLeft="@id/textPasswordReg"/>
```

Slika 12: Gradnik TextView

Iz slike razberemo lastnosti gradnika TextViev. Vidimo, da je tekst omenjenega gradnika »Ponovi geslo«, postavitev je »5dp« pod gradnikom z ID-jem »textPasswordReg« in ID letega je »textView6«.

Slika 13 predstavlja zaslonsko masko aktivnosti »registracija«.



Slika 13: Zaslonska maska aktivnosti »registracija«

3.3.1.3 Aktivnost osnovna

Zaslonska maska aktivnosti »osnovna« se prikaže, ko je na aktivnosti »main« pritisnjen gumb »Prijava«. Je torej druga v vrsti obveznih aktivnosti. Vsebuje 8 gradnikov:

- 4 krat TextView
    - Uporabnik (prikaže ime prijavljenega uporabnika)
    - Vstop
    - Izstop
    - Datum
- 2 krat AutoCompleteTextView (vnosno polje, ki glede na vnos črk podaja predloge)
    - Vstopna postaja
    - Izstopna postaja
- 2 krat Button
    - Nastavi datum
    - Poišči

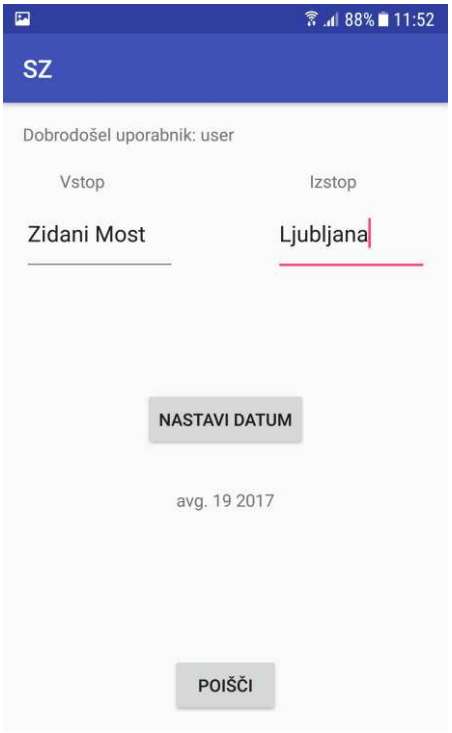
Slika 14 predstavlja programsko kodo, ki definira gradnik AutoCompleteTextView

```
<AutoCompleteTextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/acVstopna"
    android:layout_below="@+id/textView7"
    android:hint="Vstopna postaja"
    android:layout_toLeftOf="@+id/text_nastaviDatum"
    android:layout_toStartOf="@+id/text_nastaviDatum"
    android:focusable="true"
    android:focusableInTouchMode="true">
<requestFocus />
</AutoCompleteTextView>
```

Slika 14: Gradnik AutoCompleteTextView

Iz zgornje kode pa se da razbrati, da je ID samega gradnika »acVstopna«, besedilo oz. namig, kaj vnesti v polje, ki je prikazano pred uporabnikovim posredovanjem, je »Vstopna postaja« in s pomočjo vrstice »<requestFocus />« je na omenjen gradnik nastavljen fokus, kar pomeni, da bo omenjen gradnik pri odprtju te aktivnosti prvi zahteval vnos.

Torej, omenjena aktivnost je namenjena pridobivanju podatkov, ki so potrebni za nadaljnjo uporabo v aplikaciji. Uporabnik mora v primerna polja vnesti podatke o vstopni in izstopni postaji, nato izbrati datum ter pritisniti na gumb »Poišči«, s čimer pa se prikaže naslednja aktivnost. Zaslonska maska aktivnosti je prikazana na sliki 15.



Slika 15: Zaslonska maska aktivnosti »osnovna«

3.3.1.4 Aktivnost vozni_red

Je naslednja v vrsti obveznih aktivnosti, namenjena pa je, da s pomočjo prej pridobljenih podatkov v grafični obliki prikaže vse možne vozne rede. Sestavljajo jo 3 gradniki:

- 1 krat ListView (seznam)
- 2 krat TextView
    - Tekst Relacija
    - Prikaz relacije

Slika 16 predstavlja programsko kodo, ki definira gradnik ListView.

```
<ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="70dp"
    android:id="@+id/listview1"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
```

Slika 16: Gradnik ListView

Razberemo lahko, da je njegov ID »listview1« in da se nahaja »70dp« pod samim vrhom aktivnosti. Omenjen gradnik pa je dejansko namenjen prikazovanju več elementov hkrati, v našem primeru vsak element ponuja tudi možnost, da se ob pritisku nanj odpre nova aktivnost. Slika 17 predstavlja zaslonsko masko opisane aktivnosti.

Slika 17: Zaslonska maska aktivnosti »vozni_red«

Kot vidimo, vsak element gradnika ListView vsebuje podatke o prihodu in odhodu ter tip vlaka, ki so prikazani na uporabniku berljiv način.

3.3.1.5 Aktivnost rezervacija_sedeza

»Rezervacija_sedeza« je še zadnja – prav tako obvezna – aktivnost. Osnovni namen je prikaz nekaterih prej vnesenih podatkov in pa s pomočjo programske kode tudi prikaz podatkov, pridobljenih iz podatkovne baze na strežniku. Sestavljena je iz 14 gradnikov:

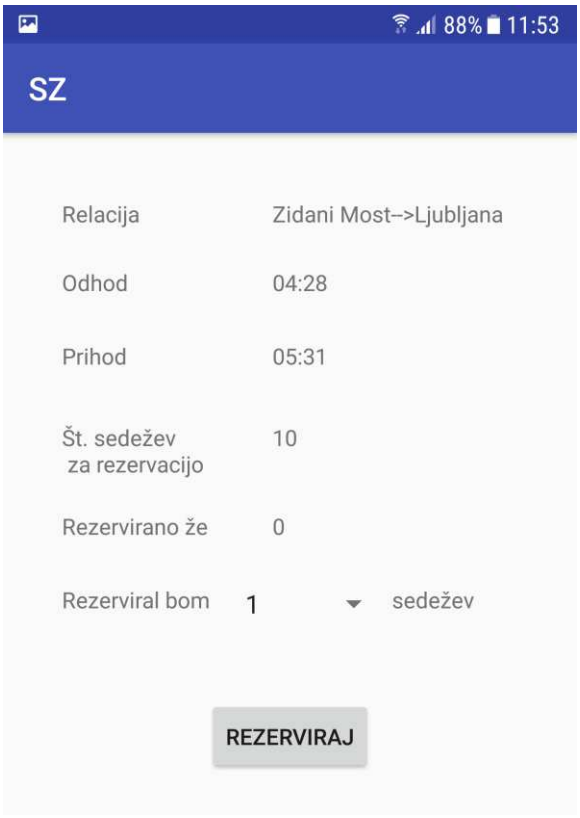
- 12 krat TextView
    - Tekst Relacija
    - Tekst Odhod
    - Tekst Prihod
    - Tekst Št. sedežev za rezervacijo
    - Tekst Rezervirano že
    - Tekst Rezerviral bom
    - Tekst Sedežev
    - Prikaz relacije
    - Prikaz odhoda vlaka
    - Prikaz prihoda vlaka
    - Prikaz števila vseh sedežev za rezervacijo
    - Prikaz števila že rezerviranih sedežev
- 1 krat Button
    - Rezerviraj
- 1 krat Spinner
    - Število sedežev

Slika 18 predstavlja programsko kodo, ki definira gradnik Spinner.

```
<Spinner
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:id="@+id/spinner_sedez"
    android:layout_alignTop="@+id/textView9"
    android:layout_toRightOf="@+id/textView13"
    android:layout_toEndOf="@+id/textView13"
    android:dropDownWidth="50dp"
    android:layout_marginLeft="18dp"
    android:layout_marginStart="18dp" />
```

Slika 18: Gradnik Spinner

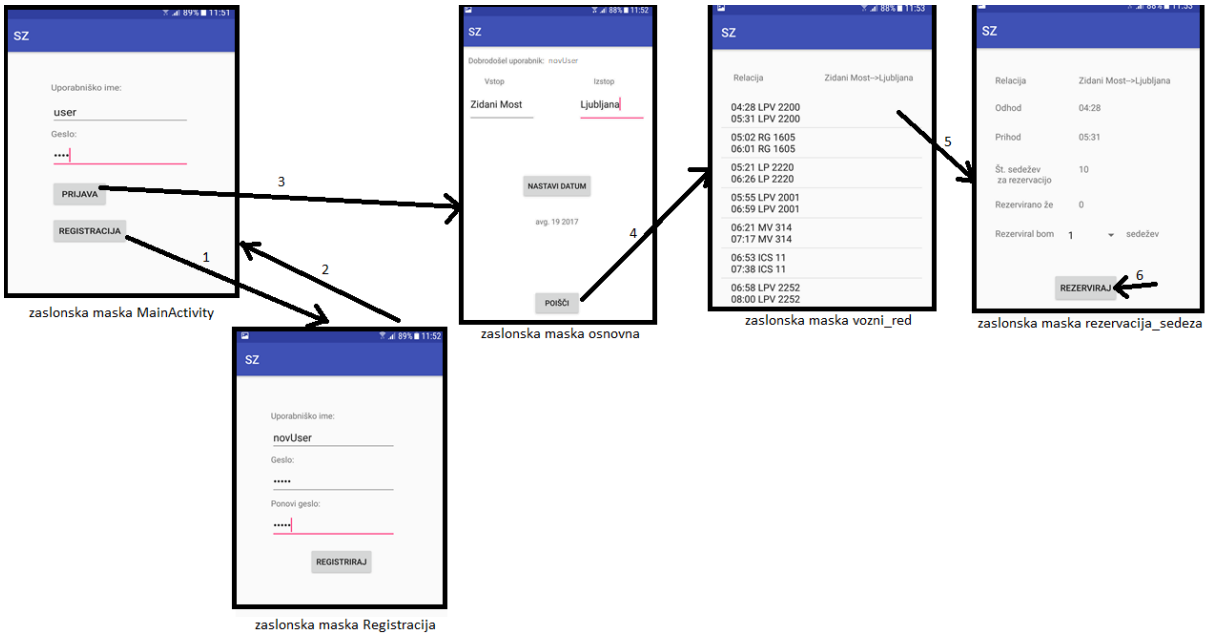
Iz zgornje kode pa lahko ponovno razberemo nekaj osnovnih lastnosti omenjenega gradnika, kot sta ID, ki je »spinner_sedez«, in širina samega gradnika »100dp«. Spinner je gradnik, ki omogoča izbiro ene vrednosti iz nabora vnaprej določenih in izgleda kot seznam. Slika 19 predstavlja grafični vmesnik omenjene aktivnosti.



Slika 19: Zaslonska maska aktivnosti »rezervacija_sedeza«

3.3.2 Tipičen scenarij uporabe aplikacije

Tipičen scenarij uporabe aplikacije prikazuje uporabo aplikacije s strani uporabnika, ki še nima svojega uporabniškega računa. Ko uporabnik zažene mobilno aplikacijo, se mu ponudi zaslonska maska »MainActivity«, ki omogoča prijavo in registracijo uporabnika. Ker uporabnik ne pozna podatkov za prijavo, izbere možnost registracije s pritiskom na gumb »Registracija« (1). Odpre se mu nova zaslonska maska »Registracija«. Uporabnik vpiše potrebne podatke (uporabniško ime, geslo in potrditev gesla) in pritisne gumb »Registriraj«. Zopet se mu odpre prva zaslonska maska (2), kjer nato vpiše podatke, ki jih je vnesel pri registraciji in se nato prijavi s pomočjo gumba »Prijava« (3). Prikaže se zaslonska maska »osnovna« in uporabnik vnese podatke o vstopni ter izstopni postaji in nastavi datum, za katerega želi, da se mu prikažejo vozni redi. Nato določi vstopno in izstopno postajo. Uporabnik želi prikaz voznih redov za trenutni dan, torej mu datuma ni potrebno spreminjati, saj aplikacija samodejno nastavi trenutni datum uporabe aplikacije. S pomočjo gumba »Poišči« se mu na zaslonski maski »vozni_red« prikažejo vozni redi za prej vnesene podatke (4). Uporabnik želi rezervirati sedež na vlaku, ki ima odhod ob določenem času. To naredi tako, da s pritiskom izbere želen vozni red in odpre se mu nova zaslonska maska »rezervacija_sedeza« (5). Tukaj ima uporabnik na voljo podatke o relaciji ter odhodu in prihodu vlaka. Prav tako lahko razbere, koliko sedežev za rezervacijo je na voljo in koliko je že rezerviranih. Da opravi rezervacijo, mora pritisniti na gradnik, ki se nahaja poleg napisa »Rezerviral bom«. Nato izbere želeno število sedežev in rezervacijo potrdi s pritiskom na gumb »Rezerviraj« (6). Slika 20 predstavlja tipičen scenarij uporabe aplikacije.



Slika 20: Tipičen scenarij uporabe aplikacije

3.3.3 Opis razredov

Pri razvoju Android aplikacij je razred predstavljen kot datoteka s končnico .java. Vsi razredi, ki v aplikaciji predstavljajo aktivnost, spadajo pod razred android.app.Activity. V posameznem razredu je treba definirati življenjski cikel aktivnosti in definirati, katera .xml datoteka se bo uporabila za prikaz grafičnega vmesnika. S pomočjo programske kode se nato lahko manipulira z gradniki na posamezni aktivnosti in piše metode oz. funkcije, ki skrbijo za delovanje aplikacije.

V nadaljevanju so opisani vsi razredi, ki jih naša mobilna Android aplikacija vsebuje, prav tako je s pomočjo slik obrazloženo, kaj predstavljajo nekateri deli programske kode.

3.3.3.1 Razred MainActivity

V tem razredu je programska koda, ki v ozadju skrbi za obdelavo podatkov, prejetih s strani aktivnosti »main«. V primeru pritiska na gumb »Prijava« se izvede del kode, ki kliče spletno storitev, zadolženo za izvedbo poizvedbe v podatkovni bazi, s katero se preveri, ali uporabnik z vnesenimi podatki (uporabniško ime, geslo) obstaja. Če poizvedba vrne pozitiven rezultat, se zažene aktivnost »osnovna«, v nasprotnem primeru pa se prikaže obvestilo (gradnik Toast), da vneseni podatki niso pravilni. Druga možnost pa je pritisk gumba »Registracija«, ki pa povzroči, da se zopet zažene nova aktivnost (»registracija«). Koda na sliki 21 predstavlja deklaracijo in inicializacijo nekaterih spremenljivk, uporabljenih v tem razredu. Na primer spremenljivki »LOGIN_URL« se nastavi vrednost, ki predstavlja URL naslov strežnika (46.101.xxx.xxx) in ime spletne storitve (prijava.php), ki skrbi za povezavo s podatkovno bazo ter izvajanje poizvedb na njej.

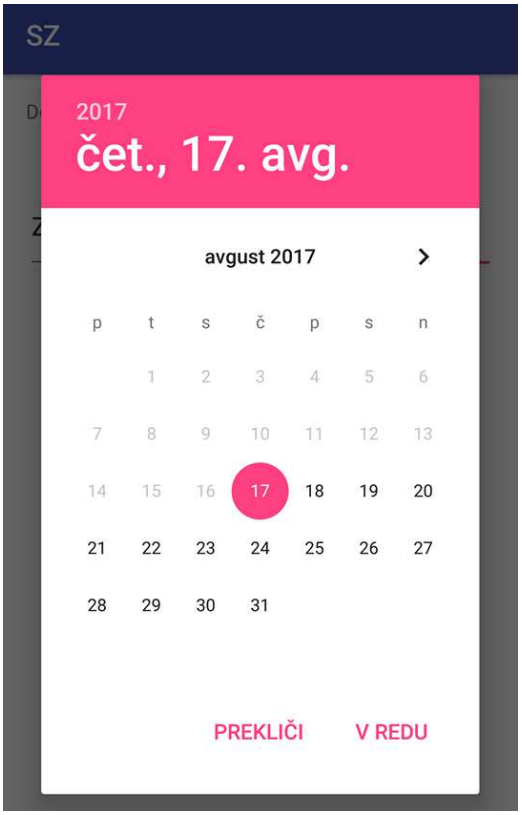
```
EditText user, pass;
String upIme, geslo;
public static final String USER_NAME = "USER_NAME";
private static final String LOGIN_URL = "http://46.101.[redacted]/prijava.php";
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    user = (EditText) findViewById(R.id.textUsername);
    pass = (EditText) findViewById(R.id.textPassword);
}
```

Slika 21: Deklaracija in inicializacija spremenljivk

3.3.3.2 Razred Osnovna

Razred »Osnovna« skrbi za aktivnost »osnovna«. Pri zagonu omenjene aktivnosti ta razred najprej poskrbi za deklaracijo vseh spremenljivk in pozdravi uporabnika z izpisom »Pozdravljen uporabnik: + uporabniško ime uporabnika«, kar prikazuje slika 15. V ozadju se izvede koda, ki napolni oba gradnika (vpis vstopne in izstopne postaje) z vsemi možnimi železniškimi postajami, datum pa se nastavi na trenutni datum v obliki »MM dd yyyy« (avg. 17 2017). Omenjena oblika je uporabljena zaradi lažje obdelave datuma v programski kodi. Ko uporabnik v katerokoli polje vnese vsaj dve črki železniške postaje, se mu samodejno ponudijo vse možne postaje, ki se začnejo s tem nizom črk (npr. ko uporabnik vpiše »LI«, mu aplikacija ponudi možnosti »Limbuš«, »Litostroj«, »Litija« …). Datum se spremeni s klikom na gumb »Nastavi datum«. To povzroči izvedbo kode, ki ponudi dodaten gradnik, imenovan DatePicker, preko katerega se lahko nato izbere datum in ga potrdi s klikom na »V redu«. Po pritisku na gumb »Poišči« se izvede koda, ki odpre aktivnost »vozni_red«. Slika 22 prikazuje gradnik DatePicker.



Slika 22: Gradnik DatePicker

3.3.3.3 Razred Registracija

Razred »Registracija«, kot že omenjeno, pride v veljavo, ko je na aktivnosti »main« pritisnjen gumb »Registracija«. Zažene se aktivnost »registracija« in prikažejo se vsi gradniki, ki jih omenjena aktivnost vsebuje. Te gradnike prikazuje slika 13. Pritisk gumba »Registriraj« povzroči, da se izvede del kode, ki kliče spletno storitev »registracija.php«, s pomočjo katere se v primeru pravilnega vnosa vseh podatkov izvede vnos podatkov v podatkovno bazo. Uporabnik je tako ustvarjen, trenutna aktivnost se zapre in odpre prejšnja aktivnosti »main«, na kateri se lahko uporabnik sedaj prijavi. Slika 23 prikazuje metodo »Registracija«, ki preverja vnos podatkov v vnosna polja. V primeru, da je vneseno uporabniško ime ali geslo krajše od treh znakov ali pa da ponovitveno geslo ni isto originalnemu, program uporabniku javi obvestilo o napaki. Če pa so podatki pravilno vneseni, se požene metoda »register« in sproži proces v razredu »Service«, ki omogoči vnos podatkov v podatkovno bazo.

```
public void Registracija(View view)
{
    upIme=username.getText().toString();
    geslo=password.getText().toString();
    geslo2=password2.getText().toString();
    if(upIme.length() > 3) {
        if(geslo.length() > 3) {
            if (geslo.equals(geslo2)) {
                register(upIme, geslo);
            } else
                Toast.makeText(Registracija.this, "Vnešeni gesli nista bili enaki!", Toast.LENGTH_LONG).show();
        } else Toast.makeText(Registracija.this, "Geslo je prekratko!", Toast.LENGTH_LONG).show();
    }else Toast.makeText(Registracija.this, "Uporabniško ime je prekratko!", Toast.LENGTH_LONG).show();

    finish();
}
```

Slika 23: Programska koda za preverjanje vnosa podatkov

3.3.3.4 Razred Service

Razred »Service« se neposredno ne navezuje na nobeno aktivnost in nima svojega grafičnega vmesnika. Namen tega razreda je s pomočjo programske kode ustvariti povezavo s strežnikom in generirati POST zahtevo s podatki, ki so kasneje potrebni za upravljanje s podatkovno bazo. Omenjeni razred je torej potreben v več primerih, kot so registracija, prijava in rezervacija sedežev. Slika 24 prikazuje ustvarjanje povezave s strežnikom in zahtevano PHP datoteko (npr. »http://46.101.xxx.xxx/prijava.php«). Razberemo lahko, da se uporabi metoda POST, zaradi česar je treba vse podatke spraviti v ustrezno obliko, za kar poskrbi metoda »getPostDataString«. Program preveri, ali je povezava uspela in pošlje POST zahtevo na spletno storitev, ta pa vrne rezultat za nadaljnjo obdelavo.

```
public String sendPostRequest(String requestURL,
                              HashMap<String, String> postDataParams) {

    URL url;
    String response = "";
    try {
        url = new URL(requestURL);

        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(15000);
        conn.setConnectTimeout(15000);
        conn.setRequestMethod("POST");
        conn.setDoInput(true);
        conn.setDoOutput(true);


        OutputStream os = conn.getOutputStream();
        BufferedWriter writer = new BufferedWriter(
                new OutputStreamWriter(os, "UTF-8"));
        writer.write(getPostDataString(postDataParams));

        writer.flush();
        writer.close();
        os.close();
        int responseCode=conn.getResponseCode();

        if (responseCode == HttpsURLConnection.HTTP_OK) {
            BufferedReader br=new BufferedReader(new InputStreamReader(conn.getInputStream()));
            response = br.readLine();
        }
        else {
            response="Error Registering";
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return response;
}
```

Slika 24: Pošiljanje POST zahteve

3.3.3.5 Razred Vozni_red

Razred »Vozni_red« prične z izvajanjem, ko je na aktivnosti »osnovna« pritisnjen gumb »Poišči«. Še pred prikazom grafičnega vmesnika se v ozadju nastavi vrednost za relacijo v obliki »vstopna postaja → izstopna postaja«, nato pa se izvede koda, ki z načinom metode HTML razčlenjevanja (ang. HTML parsing) pridobi podatke (vozne rede) iz javno dostopne spletne strani. S pridobljenimi podatki nato napolni gradnik ListView, ki dejansko predstavlja vozni red, iz katerega lahko uporabnik brez težav razbere, kdaj je prihod, odhod ter tip vlaka (LPV, LP, ICS). Zatem uporabnik s pritiskom izbere eno izmed ponujenih možnosti in odpre se aktivnost »rezervacija_sedeza«.

```
try {
    doc = Jsoup.connect("http://www.slo-zeleznice.si/sl/component/sz_timetable/?vs="+vstS+"&vi=&iz="+izsS+"&da="+datum+"T00%3A00%3A00&").get();
    Log.d("povezava", "uspešna");
    odhodi = doc.select("td.tdTime.thDeparture"); //odhod vlaka
    prihodi = doc.select("td.tdTime.thArrival"); //prihod vlaka
    tipVlak = doc.select("td.tdTrain");          //tip vlaka (LP, LPV)
    casVoznje = doc.select("td.innerMain:nth-last-child(2)");
    for (int j = 0; j < odhodi.size(); j++) {
        vrstica.add(odhodi.get(j).text() + " " +tipVlak.get(j*2).text()+"\n" + prihodi.get(j).text()+" "+tipVlak.get(j*2+1).text());
        ID.add(vstopna+","+izstopna+","+odhodi.get(j).text()+","+prihodi.get(j).text()+","+tipVlak.get(j*2).text());
    }
```

Slika 25: HTML razčlenjevanje

Slika 25 predstavlja programsko kodo HTML razčlenjevanja. Uporabljen je JSOUP, ki je Java knjižnica za delo s HTML-jem. Ponuja zelo dober API, saj z uporabo DOM, CSS in jQuery metod omogoča pridobivanje in manipulacijo s podatki. Posledično je programska koda precej krajša in lažje berljiva kot v primeru uporabe drugih knjižnic, kot je JTidy, saj te uporabljajo XPath (način naslavljanja elementov), ki je manj berljiv. V konkretnem primeru je najprej generirana povezava s spletno stranjo in shranjena v spremenljivko tipa Document. URL naslov povezave je sestavljen iz osnovnega naslova spletne strani in spremenljivk, pridobljenih iz mobilne aplikacije (vstopna, izstopna postaja in datum). Na tem spletnem mestu so podatki, ki jih aplikacija potrebuje za delovanje. Nato se s pomočjo metode »select« izberejo HTML elementi (prihod, odhod in tip vlaka) in kasneje dodajo v element ListView v uporabniku berljivi obliki, kot lahko vidimo na sliki 17.

3.3.3.6 Razred Rezervacija_sedeza

Omenjen razred se izvede v primeru, ko je na aktivnosti »vozni_red« izbran en izmed prikazanih elementov (voznih redov). Vzpostavi se aktivnost »rezervacija_sedeža«, kar prikazuje slika 19, in v ozadju se izvede programska koda. Omenjena koda, glede na prej vnesene podatke, nastavi vrednosti za relacijo, odhod in prihod. Nato pa s pomočjo razreda »Service« iz podatkovne baze pridobi ter nastavi vrednosti, ki povedo, koliko je vseh možnih sedežev za rezervacijo (glede na to vrednost se tudi napolni gradnik Spinner) in koliko sedežev je trenutno rezerviranih.

Sedaj lahko uporabnik izbere število sedežev, ki bi jih rezerviral, in to potrdi s pritiskom na gumb »Rezerviraj«. Nato se zopet kliče razred »Service« in se, glede na pridobljen rezultat, na zaslonu prikaže obvestilo o uspešni (»Rezervacija opravljena«) oz. neuspešni rezervaciji (»Rezervacija ni bila uspešna«).

Slika 26 predstavlja programsko kodo, ki definira asinhron razred. Omogoča izvajanje operacij v ozadju (imajo nižjo prioriteto) ter s tem omogoča, da se v primeru neuspele izvedbe ne poruši glavna nit (grafični vmesnik). V konkretnem primeru je razred potreben, zaradi izvedbe spletnih operacij (ang. network operations), pri katerih pogosto prihaja do

prekinitev/napak. S pomočjo metode »doInBackground (String … params)« se izvede spletna operacija in klic metode iz razreda »Service«, ki nato vrne rezultat poizvedbe. Rezultat je prikazan v tekstovni obliki, z njegovo pomočjo pa aplikacija izvaja nadaljnje operacije.

```
class Insert extends AsyncTask<String, Void, String>
{

    @Override
    protected String doInBackground(String... params) {
        int sestevekSedezev = Integer.parseInt(rezultat)+Integer.parseInt(vrednost);
        if(sestevekSedezev > Integer.parseInt(result)) return "successError";
        else {
            Service rc = new Service();
            String r = rc.sendPostRequest(URL_INSERT, podatki);
            return r;
        }
    }
```

Slika 26: Asinhron razred, namenjen izvajanju spletnih operacij

3.4 Strežnik

Na splošno je strežnik računalniški program ali naprava, ki zagotavlja funkcionalnosti ostalim programom ali napravam, ki se imenujejo odjemalci. Omenjena arhitektura se imenuje model odjemalec-strežnik. Strežniki nudijo svoje storitve, kot je deljenje virov in podatkov med odjemalci. En strežnik lahko služi več odjemalcem in en odjemalec lahko uporablja več strežnikov. Strežniki se pojavljajo v različnih vlogah: podatkovni strežnik, aplikacijski strežnik in datotečni strežnik.

Za izdelavo diplomske naloge smo za dostop in upravljanje s strežnikom (fizična naprava) uporabili programsko opremo Putty. Strežnik je bil ustrezno zaščiten z uporabniškim imenom in geslom, na njem pa je bil nameščen operacijski sistem Ubuntu različice 16.04.2 LTS. V našem primeru imamo fizični računalnik, na katerem tečeta dva strežnika (programa). Podatkovni strežnik skrbi za podatkovno bazo, medtem ko spletni strežnik skrbi za izvajanje spletnih storitev. Slika 27 prikazuje procesorske lastnosti, slika 28 pa lastnosti RAMa na našem fizičnem računalniku.

```
root@ubuntu-512mb-fra1-01:~# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list:   0
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2650L v3 @ 1.80GHz
Stepping:              2
CPU MHz:               1799.998
BogoMIPS:              3599.99
Virtualization:        VT-x
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0
```

Slika 27: Procesorske lastnosti strežnika

```
root@ubuntu-512mb-fra1-01:~# free
              total        used        free      shared  buff/cache   available
Mem:         500064      198636       19600       14656      281828      255244
Swap:             0           0           0
```

Slika 28: Lastnosti RAM-a na strežniku

Za upravljanje strežnika je bilo treba uporabiti Linux ukaze, nekateri izmed pogostejših so:

- mv (premik datotek),
- rm (brisanje datotek),
- cd (spreminjanje direktorija),
- chmod (spreminjanje dovoljenj na datoteki),
- sudo (zagon ukazov z administratorskimi pravicami).

Ker Linux distribucija na našem strežniku ni vključevala programske opreme za razvoj spletnih aplikacij, smo za to morali poskrbeti sami. Namestili smo programsko opremo LAMP. Za namestitev smo uporabili naslednje ukaze:

- Apache: sudo apt-get install apache2,

- MySQL: sudo apt-get install mysql-server,
- PHP: sudo apt-get install php5 libapache.

Za spreminjanje Apache konfiguracije smo uporabljali uporabniku zelo prijazen urejevalnik teksta (GNU nano), ki se ga požene s pomočjo ukaza: nano. Apache konfiguracijska datoteka je sestavljena iz treh sklopov, ki jih sestavljajo konfiguracija za strežniški Apache proces, konfiguracija za privzeti strežnik in konfiguracija navideznih gostiteljev. Večkrat je bilo treba izvesti ponoven zagon storitev, da so spremembe stopile v veljavo, kar smo naredili s pomočjo ukaza: sudo /etc/init.d/apache2 restart. Potrebno je bilo tudi testirati delovanje podatkovne baze, pri čemer smo si pomagali z naslednjimi ukazi: mysql -u root -p (prijava v mysql), use (izbira baze) ter describe (prikaže strukturo podane tabele).

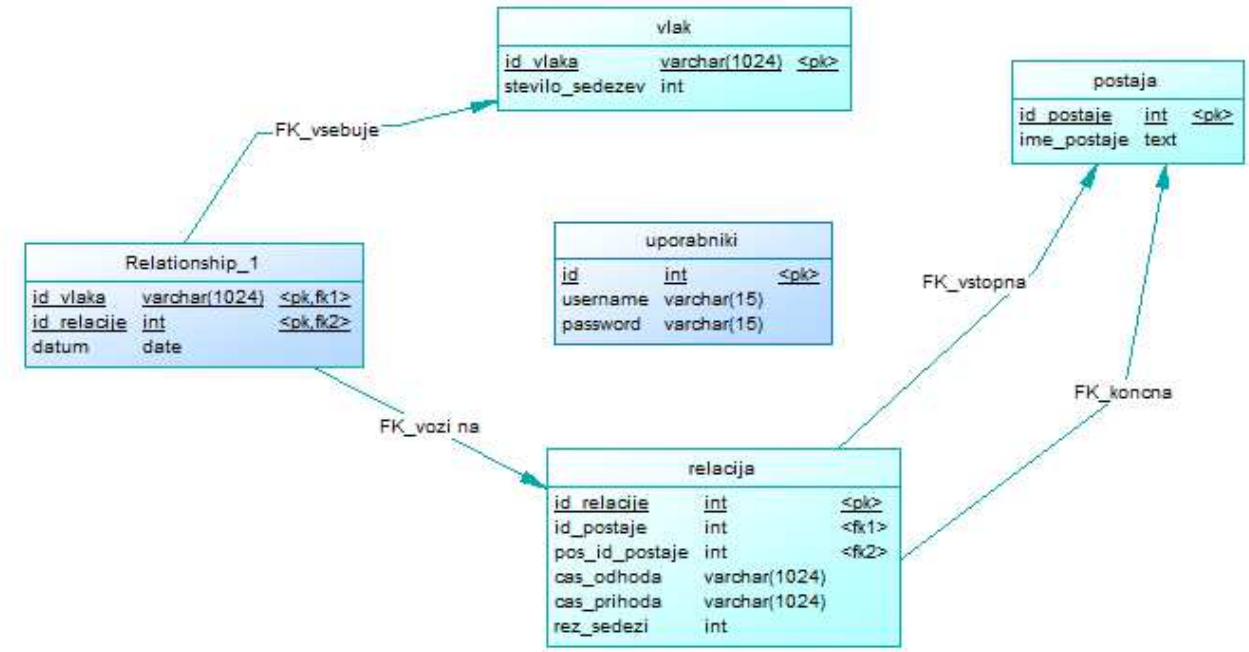
3.4.1 Podatkovna baza

Za potrebe delovanja mobilne aplikacije smo ustvarili podatkovno bazo. S pomočjo orodja PowerDesigner je bil najprej narejen logični podatkovni model (entiteta, atribut, relacija, primarni in tuji ključ), nato pa smo generirali SQL kodo (fizični podatkovni model) in jo v orodju MySQL Workbench tudi pognali ter na našem strežniku ustvarili podatkovno bazo.

Uporabljena podatkovna baza se imenuje »baza«. Vsebuje 5 tabel z naslednjimi atributi:

- vlak (id_vlaka, stevilo_sedezev),
- postaja (id_postaje, ime_postaje),
- uporabniki (id, username, password),
- relacija (id_relacije, id_postaje, pos_id_postaje, cas_odhoda, cas_prihoda, rez_sedezi),
- relationship_1 (id_vlaka, id_relacije, datum).

Slika 29 prikazuje logični podatkovni model podatkovne baze, izdelan v orodju PowerDesigner.



Slika 29: Logični podatkovni model

Slika 30 prikazuje izsek kode SQL, ki generira tabelo »postaja«.

```
/*==============================================================*/
/* Table: postaja                                               */
/*==============================================================*/
create table postaja
(
   id_postaje           int not null,
   ime_postaje          text not null,
   primary key (id_postaje)
);
```

Slika 30: Generiranje tabele s pomočjo kode SQL

Nad podatkovno bazo smo izvajali ukaze SELECT, UPDATE in INSERT. Podatke, ki jih je bilo treba brati, dodati ali spreminjati, smo v programskem jeziku Java najprej pretvorili v ustrezno obliko in šele nato napisali poizvedbe SQL v spletnih storitvah.

Slika 31 prikazuje primer poizvedbe z ukazom SELECT, ki je namenjen pridobivanju podatkov iz podatkovne baze. Poizvedba se nahaja v spletni storitvi Prijava.php in preveri, ali v podatkovni bazi obstaja uporabnik z vpisanim uporabniškim imenom in geslom. Če

poizvedba ne vrne rezultata, mobilna aplikacija javi obvestilo »Vneseni podatki niso pravilni«, v nasprotnem primeru pa aplikacija sproži prijavo in odpre se nova aktivnost.

```
$sql_query ="SELECT * FROM uporabniki WHERE username like '$username' and password like '$password';";
```

Slika 31: Stavek Select

Slika 32 prikazuje primer stavka SQL z ukazom INSERT. Omenjen ukaz je namenjen vstavljanju podatkov v podatkovno bazo. Prikazan stavek se nahaja v spletni storitvi Rezervacija.php ter je eden izmed mnogih stavkov te vrste, skrbi pa za vnos podatkov v tabelo Relationship_1, kjer se hranijo podatki, kot so: datum, tip vlaka in številka relacije. V primeru uspešno izvedenega stavka nato spletna storitev vrne status »OK«, nakar s pomočjo tega rezultata mobilna aplikacija izvede nadaljnje operacije.

```
$sql_query2 ="insert into Relationship_1 values ('$tipVlaka','$ID','$datum');";
```

Slika 32: Stavek Insert

Slika 33 prikazuje primer stavka SQL z ukazom UPDATE. Ta ukaz je namenjen spreminjanju obstoječih podatkov v podatkovni bazi. Prednost tega ukaza je, da se z njegovo pomočjo ohranja velikost podatkovne baze in se tako ne porablja dodatnega glavnega pomnilnika strežnika. Spodnji stavek ustrezno poveča atribut »rez_sedezi« v tabeli »relacija«, ki hrani podatke, kot so: številka relacije, številka vstopne ter izstopne postaje, čas odhoda ter prihoda vlaka in število rezerviranih sedežev. Spletna storitev nato zopet vrne ustrezen rezultat, glede na uspešnost poizvedbe.

```
$sql_update="update relacija set rez_sedezi = rez_sedezi + $sedezi  where id_relacije = '$eko';";
```

Slika 33: Stavek Update

3.4.2 Spletne storitve

Kot že prej omenjeno, smo morali, za potrebe komunikacije med mobilno aplikacijo in podatkovno bazo, razviti spletne storitve, ki predstavljajo most med njima ter omogočajo medsebojno komunikacijo. Storitve so bile napisane v programskem jeziku PHP in vsebujejo kodo, ki omogoča povezavo in delo s podatkovno bazo.

Izvajajo se lahko na strežniku v privzetem direktoriju /var/www/html, kar pomeni, da bi preko spletnega brskalnika do njih dostopali z URL naslovom http://46.101.XXX.XXX/prijava.php.

Pri pisanju PHP datotek je pogosto prihajalo predvsem do sintaktičnih napak – predstavljajo kršenje skladenj programskega jezika. Te smo potem odkrivali in odpravljali s pomočjo ukazov, prikazanih na sliki 34.

```
ini_set('display_errors', 1);
error_reporting(~0);
```

Slika 34: PHP ukaza za prikaz napak v skripti

Za potrebe naše aplikacije smo napisali 6 spletnih storitev:

- povezava.php (poskrbi za vzpostavitev povezave s podatkovno bazo),
- prijava.php (preveri, ali v podatkovni bazi obstaja uporabnik),
- registracija.php (doda novega uporabnika v podatkovno bazo),
- poizvedbaSedezev.php (vrne število vseh možnih sedežev za rezervacijo glede na tip vlaka),
- poizvedbaRezSedezev.php (vrne število sedežev, ki so že bili rezervirani za izbrano relacijo) in
- rezervacija.php (doda novo rezervacijo v podatkovno bazo oz. posodobi število sedežev, če rezervacija že obstaja).

Slika 35 prikazuje eno izmed omenjenih PHP spletnih storitev.

```
#UPDATE
if(mysqli_num_rows($r)>0)
{
    $row=mysqli_fetch_assoc($r);
    $eko=$row['id_relacije'];

    $sql_update="update relacija set rez_sedezi = rez_sedezi + $sedezi  where id_relacije = '$eko';";

    $k=mysqli_query($con,$sql_update);

    if($k)
    {
        echo "OKOK";
    }
    else
    {
        echo "error";
    }
}
```

Slika 35: PHP spletna storitev s stavkom SQL

Zgornja slika predstavlja del spletne storitve Rezervacija.php. Prikazuje kodo, ki v primeru, da je prejšnji stavek vrnil rezultat (spremenljivka $r), izvede stavek Update nad tabelo »relacija« in tako posodobi število rezerviranih sedežev. Če je stavek uspešno izveden, spletna storitev izpiše tekst »OKOK«, v nasprotnem primeru pa »error«. Glede na izpis se nato mobilna aplikacija odzove in uporabniku javi obvestilo o uspešni ali neuspešni rezervaciji.

Poglavje 4 Testiranje

Fazi razvoja je sledila še zadnja faza testiranja. Testiranje mobilne aplikacije je aktivnost, pri kateri se testirajo funkcionalnost, uporabnost in kakovost programske opreme. Mobilna naprava, ki smo jo uporabili za testiranje mobilne aplikacije Android med razvojem, je bila mobilni telefon Samsung Galaxy S6 z operacijskim sistemom Android Nougat 7.1.2. Po končanem razvoju smo za potrebe testiranja aplikacije ustvarili testno skupino različnih uporabnikov. Testna skupina je bila sestavljena iz 7 študentov, 5 zaposlenih in 3 starejših uporabnikov.

Poznamo več vrst testiranja mobilnih aplikacij, mi smo se odločili za naslednje:

- testiranje učinkovitosti (performance testing),
- testiranje namestitve (installation testing),
- testiranje uporabnosti (usability testing).

Testiranje učinkovitosti zagotavlja rezultate glede obnašanja mobilne aplikacije v nestandardnih pogojih. Delovanje aplikacije je bilo preizkušeno, ko je bila baterija na mobilni napravi skoraj izpraznjena, v primeru slabe internetne povezave in v primeru uporabe aplikacije več uporabnikov hkrati [8]. Znano je, da lahko operacijski sistem izključi določene servise v primeru skoraj prazne baterije oz. slabe internetne povezave, ki nato onemogočijo uporabo nekaterih funkcij aplikacije. Prav tako se lahko pojavijo težave, če aplikacijo uporablja več uporabnikov hkrati, saj se v tem primeru pošilja ogromno zahtev, kar dodatno bremeni strežnik. Naša aplikacija je v vseh primerih delovala brez težav oz. je javila ustrezna obvestila ter tako prestala testiranje učinkovitosti.

Testiranje namestitve je proces, ki preveri, ali namestitev aplikacije na mobilno napravo poteka brez težav [12]. To testiranje smo izvedli na različnih mobilnih napravah, uporabljeni pa sta bili dve različici operacijskega sistema Android. Naprave, ki so bile uporabljene, so bile

naslednje: Samsung Galaxy S6, Samsung Galaxy Xcover 4 in Samsung Galaxy Xcover 3. Aplikacija pa je bila testirana na različici sistema Android 7.0 in Android 5.1.1 Postopek namestitve in odstranitve je povsod potekal brez težav in je bil uspešen, tako da je aplikacija prestala drugi test.

Testiranje uporabnosti zagotavlja, da aplikacija dosega zadane cilje in je pozitivno sprejeta s strani uporabnikov. Naši cilji so bili, da ponudimo uporabnikom aplikacijo, ki bo zagotavljala osnovne funkcije za rezervacijo sedeža na vlaku. Študentje in zaposleni so se pozitivno odzvali predvsem na hitrost delovanja aplikacije, medtem ko so starejši uporabniki pohvalili poenostavljen način prikaza voznih redov. Vse skupine so poudarile preprost grafični vmesnik celotne aplikacije. Negativni odzivi so se nanašali na pomanjkljivo funkcionalnost rezervacije, saj aplikacija ne ponuja možnosti rezervacije točno določenega sedeža, na kar so nas opozorili študenti in zaposleni. Odzivi testiranja uporabnosti so bili tako pozitivni kot negativni, vendar pa so se negativni nanašali predvsem na dodatne funkcionalnosti, ki niso bile cilj našega diplomskega dela, tako da je aplikacija prestala tudi zadnji test.

Poglavje 5 Sklepne ugotovitve

V diplomskem delu je predstavljen razvoj mobilne aplikacije za rezervacijo sedežev na vlaku. Za razvoj te aplikacije smo se odločili, ker v Sloveniji še ne obstaja mobilna aplikacija, ki bi omenjeno funkcionalnost ponujala, medtem ko je v večini evropskih držav to že realizirano. To je bil tudi svojevrsten izziv, saj nismo vedeli, kako se bodo slovenski uporabniki na to odzvali.

Samo mobilno aplikacijo smo razvili s pomočjo orodja Android Studio v programskem jeziku Java in je namenjena izključno uporabi na mobilnem operacijskem sistemu Android. Poleg mobilne aplikacije smo uporabili tudi zunanji strežnik z operacijskim sistemom Ubuntu in na njem s pomočjo MySQL Workbench-a namestili podatkovno bazo za hrambo podatkov, ki jih aplikacija potrebuje za delovanje.

Testiranje smo izvedli s pomočjo testne skupine različnih uporabnikov (študenti, zaposleni, upokojenci). Opravili smo testiranje učinkovitosti, namestitve in uporabnosti. Rezultati vseh testiranj so bili uspešni, s pomočjo uporabnikov pa smo pridobili tudi ideje za izboljšave mobilne aplikacije.

Menimo, da ima mobilna aplikacija še precej možnosti za izboljšave. Lahko bi bolj dodelali grafični vmesnik in optimizirali poizvedbe, saj bi lahko ob večjemu številu zahtev prišlo do težav. Prav tako bi lahko aplikacijo razširili tudi na druge operacijske sisteme, kot sta iOS in Windows Phone. V tem primeru bi uporabili eno izmed hibridnih orodij (ang. cross platform), kot je Xamarin [20], ki so v današnjem času vedno bolj popularna. Cross platform je možnost, ki jo ponuja posamezno razvojno orodje. Omogoča namreč, da se v primeru razvoja aplikacije, ta izdela za mobilne naprave z različnimi operacijskimi sistemi. Spremenili bi lahko tudi način pridobivanja podatkov s spleta. Namesto mobilne aplikacije bi to lahko počela spletna storitev, ki bi skrbela tudi za obveščanje uporabnikov v primeru spremembe voznega reda, kar bi dodatno razbremenilo aplikacijo.

Literatura

[1] Android Authority. Android Studio tutorial for beginners [Online]. Dosegljivo 1. 9. 2017 na:

http://www.androidauthority.com/android-studio-tutorial-beginners-637572/

[2] Don Ho. Notepad++ [Online]. Dosegljivo 1. 9. 2017 na:

https://notepad-plus-plus.org/

[3] Kyle Mew. Android 5 Programming by Example. Dosegljivo 1. 9. 2017 na:

https://www.packtpub.com/packt/free-ebook/android-by-example

[4] Margaret Rouse. CASE (computer-aided software engineering) [Online]. Dosegljivo 1. 9. 2017 na:

http://searcherp.techtarget.com/definition/CASE-computer-aided-software-engineering

[5] Marziah Karch. What Is Google Android [Online]. Dosegljivo 1. 9. 2017 na:

https://www.lifewire.com/what-is-google-android-1616887

[6] Matjaž Štrancar, Simon Klemen. PHP in MySQL na spletnem strežniku Apache, druga izdaja. 2005

[7] MySQL. MySQL Workbench [Online]. Dosegljivo 15. 8. 2017 na:

https://www.mysql.com/products/workbench/

[8] Neotys. The begginer's guide to mobile performance testing [Online]. Dosegljivo 1. 9. 2017 na:

http://www.neotys.com/blog/beginners-guide-to-mobile-performance-testing/

[9] Paul Cobbaut. Linux Fundamentals [Online]. Dosegljivo 1. 9. 2017 na:

http://linux-training.be/linuxfun.pdf

[10] Simon Kendal. Object Oriented Programming using Java [Online]. Dosegljivo 1. 9. 2017 na:

http://bookboon.com/en/object-oriented-programming-using-java-ebook

[11] Simple wikipedia. PuTTy [Online]. Dosegljivo 15. 8. 2017 na:

https://simple.wikipedia.org/wiki/PuTTY

[12] Software Testing Help. Software Installation/Uninstallation Testing [Online]. Dosegljivo 1. 9. 2017 na:

http://www.softwaretestinghelp.com/software-installationuninstallation-testing/

[13] Sybase. Introducing PowerDesigner [Online]. Dosegljivo 1. 9. 2017 na:

http://infocenter-archive.sybase.com/help/index.jsp?topic=/com.sybase.stf.powerdesigner.docs_12.5.0/html/bwug/bwugp3.htm

[14] Turnkey. LAMP Stack [Online]. Dosegljivo 1. 9. 2017 na:

https://www.turnkeylinux.org/lampstack

[15] What is Lex? What is Yacc?. What is Yacc? [Online]. Dosegljivo 1. 9. 2017 na:

https://luv.asn.au/overheads/lex_yacc/index.html

[16] w3schools. PHP 5 Tutorial [Online]. Dosegljivo 1. 9. 2017 na:

https://www.w3schools.com/php/

[17] Wikipedia. Android Studio [Online]. Dosegljivo 15. 8. 2017 na:

https://en.wikipedia.org/wiki/Android_Studio

[18] Wikipedia. Linux [Online]. Dosegljivo 15. 8. 2017 na:

https://en.wikipedia.org/wiki/Linux

[19] Wikipedia. PowerDesigner [Online]. Dosegljivo 15. 8. 2017 na:

https://en.wikipedia.org/wiki/PowerDesigner

[20] Xamarin. Bulding Cross Platform Applications [Online]. Dosegljivo 5. 9. 2017 na:

https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/